REMARKS

This is a full and timely response to the final Official Action mailed **November 21, 2008** (the "Office Action" or "Action"). Reconsideration of the application in light of the above amendments and the following remarks is respectfully requested.

Request for Continued Examination (RCE):

Applicant hereby requests Continued Examination for this application and entry and consideration of this amendment consequent thereto.

Claim Status:

Claims 2-20 have previously been cancelled without prejudice or disclaimer and claims 21-39 have been previously added. By the forgoing amendment, the specification and claims 1, 30, and 32 have been amended. Thus, claims 1 and 21-39 are currently pending for further action.

35 U.S.C. § 112:

In the recent Office Action, claims 1 and 32 were objected to because of minor informalities. These claims have been carefully reviewed in light of the Examiner's comments and the necessary changes have been made. Following entry of this amendment, the objections to claims 1 and 32 may be reconsidered and withdrawn.

Prior Art

Rejections under 35 U.S.C. §103(a):

Claims 1 and 21-39 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,988,261 to Sokolov et al. (hereinafter "Sokolov") in view of U.S. Patent No. 6,014,519 to Egashira (hereinafter "Egashira"). For at least the following reasons, this rejection should be reconsidered and withdrawn.

Claim 1:

Independent claim 1 now recites:

> A method of optimizing the performance of an interpreter based runtime system, the runtime system including a virtual machine, the virtual machine adapted to run an application in the context of the runtime environment, the method comprising:
> augmenting a bytecode set of the virtual machine with semantically enriched opcodes, thereby constituting an application domain-specific virtual machine;
> optimizing the virtual machine based on semantics of the application to be run on the virtual machine, with at least a portion of the semantically enriched opcodes being specific to the application;
> performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off;
> analyzing frequently executed bytecodes and encoding the semantically enriched opcodes into interpreter action codes of the instruction set of the virtual machine to efficiently decode the frequently executed bytecodes;
> optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state, said system state being represented by at least one symbolic variable; and
> statically embedding the semantically enriched opcode to optimize execution of the interpreter-based runtime system.

Support for the amendment to claim 1 can be found in the Specification as originally filed by the Applicant at, for example, p. 11, line 13 - p. 12, line 3; p. 5, lines 1-3; and Figs. 3 and 6.

The final Office Action asserts that Sokolov teaches the claimed "semantically enriched opcodes." The Applicant's specification clearly teaches that semantically enriched opcodes incorporate information relating to one or more of hardware characteristics, an

application, and/or the relationship between the hardware and the application. In contrast, Sokolov teaches that "macro instructions" are automatically generated simply by operating on the byte code stream. (Sokolov, Figs. 2A, 2B, 3, 4; col. 5, line 55 to col. 7, line 19). For example, Sokolov teaches that no semantic context needs to be considered, but operates by simply identifying a sequence of two or more Java instructions within a Java instruction stream. (Sokolov, col. 7, line 7-10). This sequence of two or more Java instructions is then automatically replaced with a macro instruction. (Sokolov, Figs. 3 and 4, col. 7, lines 7-19). Because the "macro instructions" taught by Sokolov do not account for the semantic context within which the Java code operates, the "macro instructions" cannot be the "semantically enriched opcodes" taught by the Applicant.

Further, nowhere do Sokolov or Egashira teach or suggest an "application domain specific virtual machine." The Applicant teaches that the "application domain specific virtual machine" is created by "adaptive code set for a particular Java Virtual Machine" which is application-specific or application tuned. (Applicant's specification, p. 7, lines 28-32). This results in application domain specific virtual machine which is driven from the bottom-up from the application's static or dynamic behavior as opposed to a top-down/fixed pattern repository approach. (Applicant's specification, page 7, lines 28-32). Additionally, Sokolov or Egashira do not teach or suggest "with at least a portion of the semantically enriched opcodes being specific to the application."

In contrast, Sokolov teaches top-down/fix pattern repository (*see* Sokolov, Appendix A for an example of the fix pattern repository) which dictates the replacement of specific sequences of bytecode instructions irrespective of the application or semantic context. (Sokolov, Figs. 3 and 4, col. 7, lines 7-19). Consequently, Sokolov does not teach or suggest an "application domain specific virtual machine." Further, nowhere does Sokolov teach or

suggest "optimizing the virtual machine based on the semantics of the application to be run on the virtual machine."

Additionally, Sokolov does not teach or suggest "performing a quantitative comparison between execution time and memory space to determine effective semantically enriched opcodes." The Office Action points to the locations within Sokolov which use the term "limited resources." (Action, p. 3). Nowhere does Sokolov teach what these "limited resources" are or that any type of quantitative comparison is made between "limited resources." Consequently, the "limited resources" taught to by Sokolov cannot be reasonably construed as the claimed step of "performing a quantitative comparison between execution time and memory space to determine effective semantically enriched opcodes." Further, nowhere does Sokolov teach or suggest "encoding semantically enriched opcodes into interpreter action codes base upon the comparison."

Because Sokolov does not teach the claimed "semantically enriched opcodes," Sokolov cannot teach the claimed "encoding the semantically enriched opcodes into interpreter action codes of the instruction set of the virtual machine to efficiently decode the frequently executed bytecodes."

The Office Action also asserts that Sokolov teaches "optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state." In addition to not teaching "semantically enriched opcodes", Sokolov does not teach or suggest "interpreter action codes" or "optimization translation…according to a system state." The Applicant's specification teaches that "interpreter action codes" map the bytecode sequence in the semantically enriched opcode into optimized portable native C code for the target machine. (Applicant's specification, p. 10, lines 25-31). The virtual machine system state is

tracked using symbolic interpretation and the translation of the semantically enriched opcodes is optimized according to that system state. (Applicant's specification, p. 11, lines 13-15).

Nowhere does Sokolov teach monitoring or tracking a system state of a virtual machine and optimizing the translation of opcodes according to that system state. As stated above, Sokolov teaches that teaches top-down/fix pattern repository which dictates the replacement of specific sequences of bytecode instructions irrespective of the application or semantic context. Consequently, Sokolov does not teach or suggest "optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state."

The Office Action concedes that Sokolov does not disclose "performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off." Consequently, the Office Action cites Egashira. However, Egashira fails to remedy the shortcomings of Sokolov. Specifically, neither Sokolov nor Egashira teach or disclose "semantically enriched opcode" or "encoding semantically enriched opcodes into interpreter action codes based upon the trade-off."

The Supreme Court recently addressed the issue of obviousness in *KSR Int'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727 (2007). The Court stated that the *Graham v. John Deere Co. of Kansas City*, 383, U.S. 1 (1966), factors still control an obviousness inquiry. Under the analysis required by *Graham v. John Deere*, 383 U.S. 1 (1966) to support a rejection under § 103, the scope and content of the prior art must first be determined, followed by an assessment of the differences between the prior art and the claim at issue in view of the ordinary skill in the art. In the present case, the scope and content of the prior art, as evidenced by Sokolov and Egashira, did not include the claimed subject matter, particularly

"a semantic enriched opcode," "application domain specific virtual machine," "optimizing the virtual machine based on the semantics of the application to be run on the virtual machine," "performing a quantitative comparison between execution time and memory space to determine effective semantically enriched opcodes" or "optimizing the translation by the interpreter action codes of the semantically enriched opcodes according to a system state."

The differences between the cited prior art and the claimed subject matter are significant because the claimed method provides substantial improvements in the performance of virtual machines by leveraging semantic information. (Applicant's specification, p. 14, lines 3-20). Thus, the claimed subject matter provides features and advantages not known or available in the cited prior art. Consequently, the cited prior art will not support a rejection of claim 1 and its dependent claims under 35 U.S.C. § 103 and *Graham*.

Claim 30:

Independent claim 30 now recites:

A system for optimizing performance of an interpreter based runtime system, the runtime system including a virtual machine, the system comprising:

application code;

an embedded processor;

a virtual machine configured to translate said application code into native machine code compatible with said embedded processor;

a detection module, said detection module being configured to analyze said application code to identify code segments that could be efficiently represented as *semantically enriched opcodes*, *at least a portion of said semantically enriched opcodes being specific to said application*;

an embedding module, said embedding module being configured to *embed said semantically enriched opcodes in said application*;

a code generation module, said code generation module being configured to generate optimized action code for translating said semantically enriched opcodes *according to symbolic states, each of said symbolic states being represented by at least one symbolic variable*; and

a build module configured create an *application domain-specific virtual machine* by incorporating said optimized action code and a bytecode set comprising said semantically enriched opcodes into said virtual machine.
(Emphasis added)

Support for the amendment to claim 30 can be found in the Specification as originally filed by the Applicant at, for example, p. 11, line 13 - p. 12, line 3; p. 5, lines 1-3; and Figs. 3 and 6.

As discussed above, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "semantically enriched opcodes" or "semantically enriched opcodes being specific to the application." Further, Sokolov and Egashira do not teach or suggest "translating said semantically enriched opcodes according to symbolic states." The Applicant's specification teaches that the symbolic states are representations of the current condition of a virtual machine. (Applicant's specification, Fig. 6; p. 11, lines 13 – p. 12, line 3). By leveraging this information, more efficient opcodes can be created which significantly improve the performance of applications the virtual machine is running. Further, nowhere do Sokolov or Egashira teach or suggest the claimed "application domain specific virtual machine." In contrast, Sokolov teaches top-down/fix pattern repository approach (see Sokolov, Appendix A for an example of the fix pattern repository) which dictates the replacement of specific sequences of bytecode instructions irrespective of the application or semantic context. Consequently, Sokolov does not teach or suggest an "application domain specific virtual machine."

In addition to major shortcomings of the cited references in teaching the individual elements recited within the Applicant's claims, the cited references fail to teach a large portion of the recited relationships between the elements. "To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974)." M.P.E.P. § 2143.03.

Accord. M.P.E.P. § 706.02(j). For example, the cited references do not teach or suggest a "code generation module being configured to generate optimized action code for translating said semantically enriched opcodes according to symbolic state." Nowhere with the large portions of Sokolov cited within the Action is a "code generation module" discussed with relation to "optimized action codes," "semantically enriched opcodes," or a system "symbolic state." Further, the cited references do not teach or suggest "a build module configured to create an application domain-specific virtual machine by incorporating said optimized action code and a bytecode set comprising said semantically enriched opcodes into said virtual machine." Nowhere with the portions of Sokolov cited by the Action is a "build module" discussed with relation to "an application domain-specific virtual machine," or a "semantically enriched opcodes."

The Office Action concedes that Sokolov does not disclose "performing a quantitative trade-off between execution time and memory space to determine effective semantically enriched opcodes and encoding the semantically enriched opcodes into interpreter action codes based upon the trade-off." Consequently, the Office Action cites Egashira. However, Egashira fails to remedy the shortcomings of Sokolov. Specifically, neither Sokolov nor Egashira teach or disclose "semantically enriched opcode" or "encoding semantically enriched opcodes into interpreter action codes based upon the trade-off."

Under the analysis required by *Graham v. John Deere*, 383 U.S. 1 (1966) to support a rejection under § 103, the scope and content of the prior art must first be determined, followed by an assessment of the differences between the prior art and the claim at issue in view of the ordinary skill in the art. In the present case, the scope and content of the prior art, as evidenced by Sokolov and Egashira, did not include the claimed subject matter, particularly "a semantic enriched opcode," "application domain specific virtual machine,"

"code generation module being configured to generate optimized action code for translating said semantically enriched opcodes according to symbolic state," and "a build module configured to create an application domain-specific virtual machine by incorporating said optimized action code and a bytecode set comprising said semantically enriched opcodes into said virtual machine."

The differences between the cited prior art and the claimed subject matter are significant because the claimed method provides substantial improvements in the performance and size of virtual machines by leveraging semantic information. (Applicant's specification, p. 14, lines 3-20). Thus, the claimed subject matter provides features and advantages not known or available in the cited prior art. Consequently, the cited prior art will not support a rejection of claim 30 and its dependent claims under 35 U.S.C. § 103 and *Graham*.

Additionally, various dependent claims of the application recite subject matter that is further patentable over the cited prior art. Specific, non-exclusive examples follow.

<u>Claims 21 and 31</u>

Claim 21:

> The method of claim 1, further comprising analyzing an application code using static optimization, the static optimization comprising parsing the application code to identify at least one repeated sequence of bytecodes and replace the at least one repeated sequence of bytecodes with a semantically enriched opcode.

Similarly, claim 31 recites:

> The system of claim 30, wherein said detection module is configured to analyze said application code using static optimization, said static optimization comprising parsing said application code to identify at least one repeated sequence of bytecodes and replace said at least one repeated sequence of bytecodes with a semantically enriched opcode.

Claims 21 and 31 are patentable for at least the same reasons given above for the patentability of independent claims 1 and 30. Additionally, nowhere does Sokolov teach or suggest the recited "static optimization" of application code. In contrast, Sokolov teaches that "a Java macro instruction generator" reads and operates on "a stream of Java Bytecode ***during Java Bytecode verification***." (Sokolov, col. 4, lines 1-15). Operations on a "stream of Java Bytecode during Java Bytecode verification" cannot be reasonably interpreted as the claimed "static optimization." Consequently, the cited prior art will not support a rejection of claims 21 and 31 under 35 U.S.C. § 103 and *Graham*. For at least this additional reason, the rejection of claims 21 and 31 should be reconsidered and withdrawn.

Claims 23 and 33

Claim 23:

The method of claim 1, further comprising:
discovering at least one repetitive computational sequence used in a symbolic state; and
generating a semantically enriched opcode corresponding to the at least one repetitive computational sequence used in the symbolic state.

Similarly, claim 33 recites:
The system of claim 30, wherein said generation module is further configured to:
discover at least one repetitive computational sequence used in a said symbolic state; and
generate a semantically enriched opcode corresponding to the at least one repetitive computational sequence used within said symbolic state.

As discussed above, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "semantically enriched opcodes" or "symbolic states." Further, Sokolov and Egashira do not teach or suggest the claimed relationship between the

semantically enriched opcodes and symbolic state, namely, "semantically enriched opcode corresponding to the at least one repetitive computational sequence used within said symbolic state." Consequently, the cited prior art will not support a rejection of claims 23 and 33 and under 35 U.S.C. § 103. For at least this additional reason, the rejection of claims 23 and 33 should be reconsidered and withdrawn.

Claims 24 and 34

Claim 24:

> The method of claim 23, wherein the symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments.

Similarly, claim 34 recites:

> The system of claim 33, wherein said symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments.

As discussed above, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited a "symbolic state." Further, Sokolov and Egashira do not teach or suggest a "symbolic state comprises at least one of: control flow, data flow, entry points, and operational arguments." The Action points to a several sections within Sokolov and asserts, without further reasoning, that the text of the sections teaches the claimed subject matter. (Action p. 8). The Applicant is at a loss as to the relevance of the cited sections to the claimed subject matter. Nowhere within the cited sections is a "symbolic state" taught or suggested. Further nowhere within the cited sections is a relationship between a "symbolic state" and any of a "control flow, data flow, entry points, and operational arguments" taught or suggested. Consequently, the cited prior art will not support a rejection of claims 24 and 34

and under 35 U.S.C. § 103. For at least these additional reasons, the rejection of claims 24 and 34 should be reconsidered and withdrawn.

Claims 25 and 35

Claim 25:

> The method of claim 1, further comprising optimizing native code output by the virtual machine using a global optimizing compiler.

Similarly, claim 35 recites:

> The system of claim 30, further comprising a global optimizer compiler configured to optimizing native code output by said virtual machine.

Claims 25 and 35 are patentable for at least the same reasons given above for the patentability of claims 1 and 30. Additionally, Sokolov and Egashira do not teach or suggest, either separately or in combination, the recited "optimization of native code output by the virtual machine using a global optimizing compiler." Again, the Action points to a several sections within Sokolov and asserts, without further reasoning, that the text of the sections teaches the claimed subject matter. (Action p. 8). The Applicant is again at a loss as to the relevance of the cited sections to the claimed subject matter. Nowhere within the cited sections is a "global optimizing compiler" mentioned. The Applicant is left to assume that the Office Action misconstrued the substitution of "Java macro instruction for two or more Java Bytecode instructions" recited by Sokolov at col. 5, line 28 to col. 6, line 38 as the claimed "global optimizing compiler." One of skill in the art readily recognizes that substituting a "Java macro instruction" for "two or more Java Bytecode instructions" does not teach or disclose the operation of a global optimizing compiler on native code output by a virtual machine. Additionally, the global optimizing compiler claimed by the Applicant is

recited as acting on the output of the virtual machine, not during the verification of Bytecode. For at least these additional reasons, the rejection of claims 25 and 35 should be reconsidered and withdrawn.

Claims 27, 29, 37 and 39:

Claims 27, 29, 37 and 39 recite various limitations which related to the method for substituting a semantically enriched opcode for a corresponding code segment. Specifically, claims 27 and 37 recite modification of the virtual machine "by inserting a stub, said stub automatically loading a semantically enriched opcode when said virtual machine encounters an identified code segment." Claims 29 and 39 recite method/system in which a class loader substitutes semantically enriched opcodes for a corresponding code segment. The cited sections within Sokolov do not mention specific methods of making substitutions into computer code. Nowhere do Sokolov or Egashira teach or suggest "stubs" or "class loaders" which make substitutions for code segments. For at least these additional reasons, the rejection of claims 29 and 39 should be reconsidered and withdrawn.

Conclusion:

In view of the foregoing arguments, all claims are believed to be in condition for allowance over the prior art of record. Therefore, this response is believed to be a complete response to the Office Action. However, Applicant reserves the right to set forth further arguments in future papers supporting the patentability of any of the claims, including the separate patentability of the dependent claims not explicitly addressed herein. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed.

The absence of a reply to a specific rejection, issue or comment in the Office Action does not signify agreement with or concession of that rejection, issue or comment. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment. Further, for any instances in which the Examiner took Official Notice in the Office Action, Applicants expressly do not acquiesce to the taking of Official Notice, and respectfully request that the Examiner provide an affidavit to support the Official Notice taken in the next Office Action, as required by 37 CFR 1.104(d)(2) and MPEP § 2144.03.

If the Examiner has any comments or suggestions which could place this application in better form, the Examiner is requested to telephone the undersigned attorney at the number listed below.

Respectfully submitted,

DATE: January 21, 2009

/Steven L. Nichols/
Steven L. Nichols
Registration No. 40,326

Steven L. Nichols, Esq.
Managing Partner, Utah Office
Rader Fishman & Grauer PLLC
River Park Corporate Center One
10653 S. River Front Parkway, Suite 150
South Jordan, Utah 84095

(801) 572-8066
(801) 572-7666 (fax)